天玑大数据处理引擎关键技术及应用

查礼 程学旗

1 概述

近年来，越来越多的国内外互联网公司和传统企业都已意识到数据资产规模化带来的潜在价值。这些呈爆炸性增长的数据资产的类型以非结构化和半结构化为主，如何低成本且高效率地存储和处理 PB[1]至 EB[2]量级的数据成为业界面临的极大挑战。谷歌（Google）公司陆续提出了 MapReduce[1] 编程框架，GFS[3]文件系统[2]以及 BigTable[3] 存储系统，从而成为大数据处理技术的开拓者和领导者。而源于这三项技术的 Apache Hadoop[4]等开源项目则成为大数据处理技术的事实标准，迅速推广应用于国内外各大互联网企业，成为 PB 量级大数据处理的成熟技术和系统。天玑大数据处理引擎是构建在 Hadoop 之上的面向大数据计算 (Big Data Computing)的工具集，其中包含了很多天玑团队既有的研究成果。这些研究成果已在互联网公司实际生产系统上经受住考验，如 RCFile 已应用到 Facebook（脸书）公司，CCIndex 已应用于淘宝网的“数据魔方”，天玑 Base 已应用到腾讯“广点通”等。这些关键技术构成了天玑大数据处理引擎的核心竞争力，软件的生态环境也借由开源社区得到良性发展。

2 大数据处理技术现状

2.1 谷歌

谷歌在搜索引擎上所获得的巨大成功，很大程度上是由于采用了先进的大数据管理和处理技术，是针对搜索引擎所面临的日益膨胀的海量数据存储问题以及在此之上的海量数据处理问题而设计的。

针对内部网络数据规模超大的特点，谷歌提出了一整套基于分布式并行集群方式的基础架构技术，利用软件的能力来处理集群中经常发生的节点失效问题。谷歌使用的大数据平台包括四个相互独立又紧密结合在一起的系统：谷歌文件系统（GFS），针对谷歌应用程序的特点提出的 MapReduce 编程模式，分布式的锁机制 Chubby 以及大规模分布式数据库 BigTable。

GFS 是一个大型的分布式文件系统，它为谷歌云计算提供海量存储，并且与 Chubby、MapReduce 和 BigTable 等技术结合得十分紧密，处于系统的底层。它与传统的分布式文件系统有许多相同的目标，例如性能、可伸缩性、可靠性以及可用性。除此之外，它的设计还受到谷歌应用负载和技术环境的影响。相对于传统的分布式文件系统，为了达到成本、可靠性和性能的最佳平衡，GFS 从多个方面进行了简化：(1) 采用集中式元数据管理；(2) 不缓存数据；(3) 在用户态下实现；(4) 只提供专用接口。另外，GFS 将节点失效视为系统的常态，提供了极强的系统容错功能；设置三个数据块副本，以增强数据可靠性；使用了链式写和版本控制的双重保证数据一致性，即数据块的所有在线副本组成一条写更新链，用户进行

---

[1] 10^{15}（千万亿）字节
[2] 10^{18}（百亿亿）字节
[3] Google File System，谷歌公司为了存储海量搜索数据而设计的专用文件系统

写操作时，数据链式写入所有副本，当链上的所有副本都完成更新后，写操作才会成功，并更新对应数据块的版本号。

MapReduce 是处理海量数据的并行编程模式，用于大规模数据集的并行运算。MapReduce 通过“Map（映射）”和“Reduce（化简）”这样两个简单的概念来参加运算。用户只需要提供自己的 Map 函数以及 Reduce 函数就可以在集群上进行大规模的分布式数据处理。这一编程环境能够使程序设计人员编写大规模的并行应用程序时不用考虑集群的可靠性、可扩展性等问题。应用程序编写人员只需要将精力放在应用程序本身，关于集群的处理问题则交由平台来完成。与传统的分布式程序设计相比，MapReduce 封装了并行处理、容错处理、本地化计算、负载均衡等细节，具有简单而强大的接口。正是由于 MapReduce 具有函数式编程语言和矢量编程语言的共性，使得这种编程模式特别适合于非结构化和结构化的海量数据的搜索、挖掘、分析等应用。

Chubby 是提供粗粒度锁服务的一个文件系统，它基于松耦合分布式文件系统，解决了分布的一致性问题。这种锁只是一个建议性的锁而不是强制性的锁。通过使用 Chubby 的锁服务，用户可以确保数据操作过程中的一致性。GFS 使用 Chubby 来选取一个 GFS 主服务器，BigTable 使用 Chubby 指定一个主服务器并发现、控制与其相关的子表服务器。

大规模分布式数据库 BigTable 是基于 GFS 和 Chubby 开发的分布式存储系统。很多应用程序对于数据的组织是非常有规则的。一般来说，数据库对于处理格式化的数据还是非常方便的。但是由于关系数据库要求很强的一致性，很难将其扩展到很大的规模。为了处理谷歌内部大量的格式化以及半格式化数据，谷歌构建了弱一致性要求的大规模数据库系统 BigTable。BigTablede 在很多方面和数据库类似，但它并不是真正意义上的数据库。谷歌包括 Web 索引、卫星图像数据等在内的很多海量结构化和半结构化数据都是存储在 BigTable 中的。BigTable 的内容按照行来划分，将多个行组成一个小表（Tablet），保存到某一个服务器节点中。

2.2 Hadoop

Apache Nutch 是 Hadoop 的源头。该项目始于 2002 年，是 Apache Lucene 的子项目之一。当时的系统架构尚无法扩展到存储并处理拥有数十亿网页的网络化数据。谷歌在 2003 年在 SOSP[4]上公开了描述其分布式文件系统的论文 *The Google File System*（《谷歌文件系统》），为 Nutch 提供了及时的帮助。2004 年，Nutch 的分布式文件系统(NDFS[5])开始开发。同年，谷歌在 OSDI[6]上发表了题为 *MapReduce: Simplified Data Pro cessing on Large Clusters*（《MapReduce—简化的大规模集群数据处理》）的论文，受到启发的道.卡廷

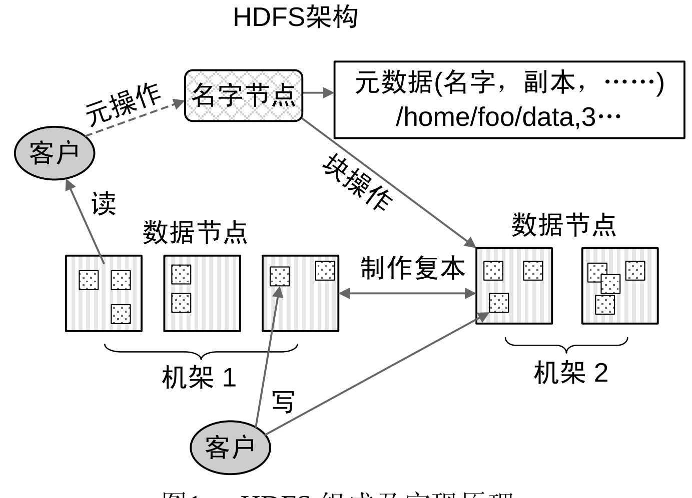


图1. HDFS 组成及实现原理

4 Symposium on Operating Systems Principles，操作系统原理会议
5 Nutch Distributed File System
6 Operating Systems Design and Implementation，操作系统设计与实现国际会议

（Doug Cutting）等人开始实现 MapReduce 计算框架，并与 NDFS（Nutch Distributed File System）结合起来，共同支持 Nutch 的主要算法。至 2006 年，这个框架逐渐成为一套完整而独立的软件，命名为 Hadoop。2008 年初，Hadoop 成为 Apache 的顶级项目，不仅用于雅虎（Yahoo! ），还在众多互联网企业得以应用。

Hadoop 核心由两部分组成：HDFS 和 MapReduce，其中 HDFS 是 Google GFS 的开源版本，是一个高可靠的分布式文件系统。它能够提供高吞吐率的数据访问能力，适合存储海量（PB 级）数据，其实现原理如图 1 所示。

HDFS 全部在用户态使用 Java 语言编写。名字节点（NameNode，也是主节点）在系统中只有一个实例，采用键值（Key-Value）全内存式管理模式，用于管理文件系统的元数据。元数据包括名字空间、副本数量及位置、文件到块的映射关系。数据节点（DataNode）负责固定大小数据块的存储(通常为 64MB)。一个文件(home/foo/data)由存储在多个数据节点上的数据块构成，客户端访问数据时经由名字节点获得数据块的存储位置，再与数据块所在的数据节点交互，写入或读出数据。

MapReduce 计算框架实现了由谷歌工程师提出的 MapReduce 编程模型，其原理如图 2 所示。

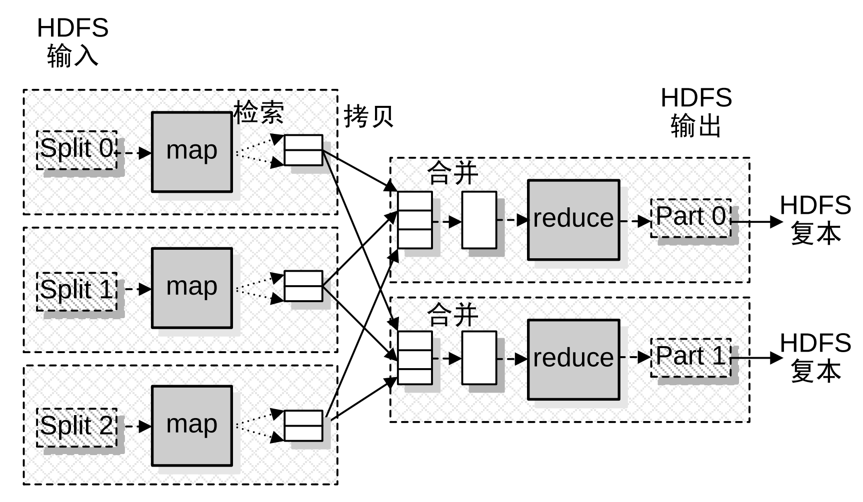


图2. MapReduce 编程模型示意图

当一个 MapReduce 作业提交给 Hadoop 集群时，相关的输入数据将首先被划分为多个片断，然后由作业跟踪程序（Job Tracker）挑选空闲的任务跟踪器（Task Tracker）对数据片断并行地执行 Map 任务。接着这些由 Map 任务产生的中间记录会被再次划分并由作业跟踪程序挑选空闲的任务跟踪器对它们并行地执行 Reduce 任务，从而获得和每个键值相对应的数据集合作为运算结果。这样的过程将被反复执行，直到 MapReduce 作业中所有的 Map 任务和 Reduce 任务执行完毕。

Pig	Chukwa	Hive	HBase
MapReduce	HDFS	Zoo Keeper	
Core	Avro		

图3. Hadoop 各组成部分之间的关系示意图

虽然在 Hadoop 中有名的是 MapReduce 及其分布式文件系统 HDFS，但还有其他子项目支持开发的工具提供配套和补充性服务。这些子项目之间的关系如图 3 所示。

- Core：一系列分布式文件系统和通用读写（I/O）的组件和接口（序列化、Java 远程调用（RPC）和持久化数据结构)
- Avro：一种提供高效、跨语言远程调用的数据序列系统，持久化数据存储
- MapReduce：分布式数据处理模式和执行环境
- HDFS：以块数据为单位存储并具有副本机制的分布式文件系统
- Pig：一种运行在 MapReduce 和 HDFS 的集群上的高层（High Level）数据流语言和运行环境，用以检索海量数据集
- HBase：一个分布式列存储数据库，使用 HDFS 作为底层存储，同时支持 MapReduce 的批式计算和点查询（随机读取）
- ZooKeeper：一个分布式高可用的协同服务，提供分布式锁相关的基本服务，用于支持分布式应用构建
- Hive：分布式数据仓库，用于管理 HDFS 中存储的数据，并提供基于 SQL 的查询语言（由运行时解释引擎转换为 MapReduce 作业）用以查询数据
- Chukwa：分布式数据收集和分析系统，用于监控大规模分布式系统并基于 HDFS 和 MapReduce 生成报告。

3 天玑大数据处理引擎

Hadoop 作为 Google 系统的开源实现已经在互联网领域得以广泛的应用。国外企业，如雅虎、Facebook、亚马逊（Amazon)、IBM 等和国内企业，如百度、中国移动、淘宝、腾讯、网易、人人网等都在使用 Hadoop 软件。Hadoop 核心以及外围工具和服务为快速构建互联网量级的数据处理提供了可直接使用的工具集。开源软件的众包特点和草根特性在 Hadoop 软件上得以充分体现。开源软件应用最广泛的是互联网公司，尤其是那些开始创业的小企业（start-ups），在技术选型方面 LAMP[7]、memcache[8]、Hadoop 是他们的软件构件首选。这里，成本是一方面的原因；另一方面，选用开源软件可以很容易地根据自身业务特点进行定制开发，形成企业的核心竞争力。

互联网企业在使用 Hadoop 的同时也根据自身业务需求，开发出相关的软件和工具，不断增强 Hadoop 软件功能和壮大 Hadoop 的开发队伍。比如 Facebook 公司因为其数据分析工程师只熟悉 SQL 语言而不熟悉 MapReduce 编程框架，由此催生 Hive 这样的项目。其初衷就是实现 SQL 到 MapReduce 的解释执行。Hive 现在已经演化为数据仓库的实用解决方案。这从一个侧面反映了软件开放源代码对信息技术的巨大推动作用。

国内的大数据计算技术和产业发展应该从开源文化中汲取经验，重视开源软件，以开源软件为基础形成核心竞争力。天玑大数据处理引擎的研发就是遵循了这一原则，发挥计算所科研能力强的优势，面向大数据计算的技术需求，解决关键问题，形成关键技术。利用开源的 Hadoop 作为平台，集成整合并回馈开源社区，从而达到天玑大数据处理引擎软件生态环境的良性循环和良性发展。

如图 4 所示，天玑大数据处理引擎的特点是：针对企业计算领域的大数据生产需求，兼容传统关系数据库操作接口，支持流式计算、图计算等模式。支持 EB 级数据分布式存储及离线式非线性处理能力，PB 级数据在线式处理能力，达到每秒千万记录级流式处理能力。

[7] 指一组通常一起使用来运行动态网站或者服务器的自由软件：Linux 操作系统 Apache（阿帕奇）网页服务器、MySQL 数据管理系统、PHP 脚本语言
[8] 一个高性能的分布式的内存对象缓存系统

达到这样的目标需要攻克统一存储、查询引擎、隔离机制、自动化运维和软硬件一体等技术难点和难题，最终逐步建立起包含模型、算法、接口、开发库等在内的天玑大数据处理引擎软件栈和生态环境。

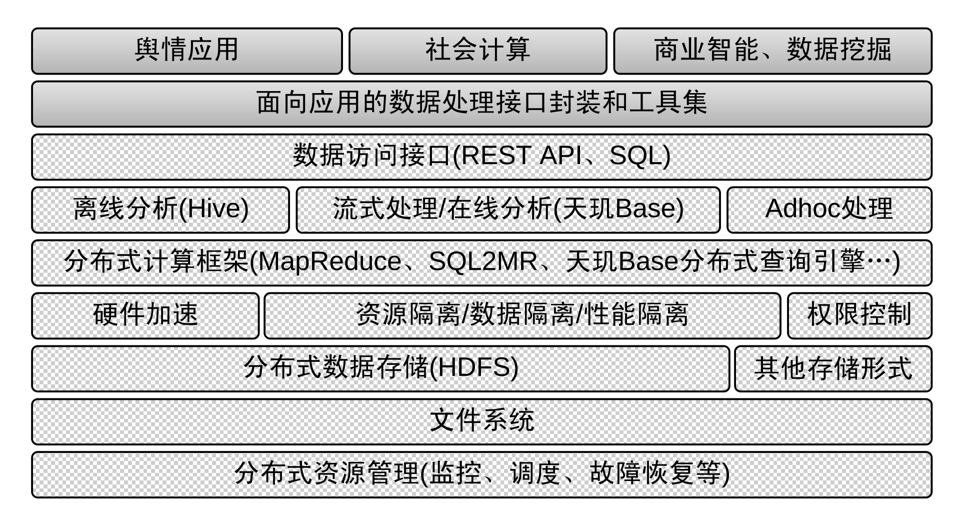


图4. 天玑大数据处理引擎逻辑组成

4 天玑大数据计算关键技术研发

国内的高校和科研院所基于 Hadoop 在数据存储、资源管理、作业调度、性能分析优化、系统高可用性和安全性等方面开展了研究工作，相关研究成果多以开源形式贡献给 Hadoop 社区。近两年，我们主要在数据的存储和索引等技术上开展研究工作，形成了天玑大数据处理引擎的核心竞争力。

4.1 行列混合式数据存储技术

Hive 是基于 Hadoop 的一个数据仓库工具。在 Hive 中，绝大部分操作都需要对数据进行存储和读取，因此 Hive 的数据存储格式及访问方式会极大地影响 Hive 的运行效率。之前，Hive 中采用 SequenceFile 文件格式来管理数据。SequenceFile 采用行存储，只能按照行存、行取的方式来访问数据，当要读取某一列时需要先取出所有数据，然后再从中提取出某一列的数据，效率很低。

RCFile 是要研究一种可以按行和列两种方式访问数据的存储模式，用以提高 Hive 访问数据的效率。RCFile 是一种以属性（列）值顺序存储记录集的数据表存储格式。该存储结构可以按查询需要读取记录中的属性数据，并支持以列数据为单位的数据压缩机制和查询运算执行方法。由于数据仓库应用中查询所需属性常常少于数据表全部属性，列存储结构可以有效提高查询处理效率。

假设有一个数据表 Relation，利用传统的二进制行存储技术进行存储，则会按照一行一行的方式将数据表中的数据存储下来，存储格式如下图（上）所示。

利用 RCFile 的存储格式进行存储则首先将记录分到不同的行分组中，然后将行进行列切片，以列顺序来储存一个行分组中的所有数据。在每个分组中，首先将所有的元数据值部分作为一个整体存起来，然后顺序存储分组内每列的数据值。

此外，RCFile 采用了逐列压缩技术，即每个行分组内每列数据单独进行压缩。这一技术的优点在于只需要对查询所需要的列进行解压处理。目的在于同时发挥数据压缩和列存储

技术的优势。存储格式如图 5（下）所示。

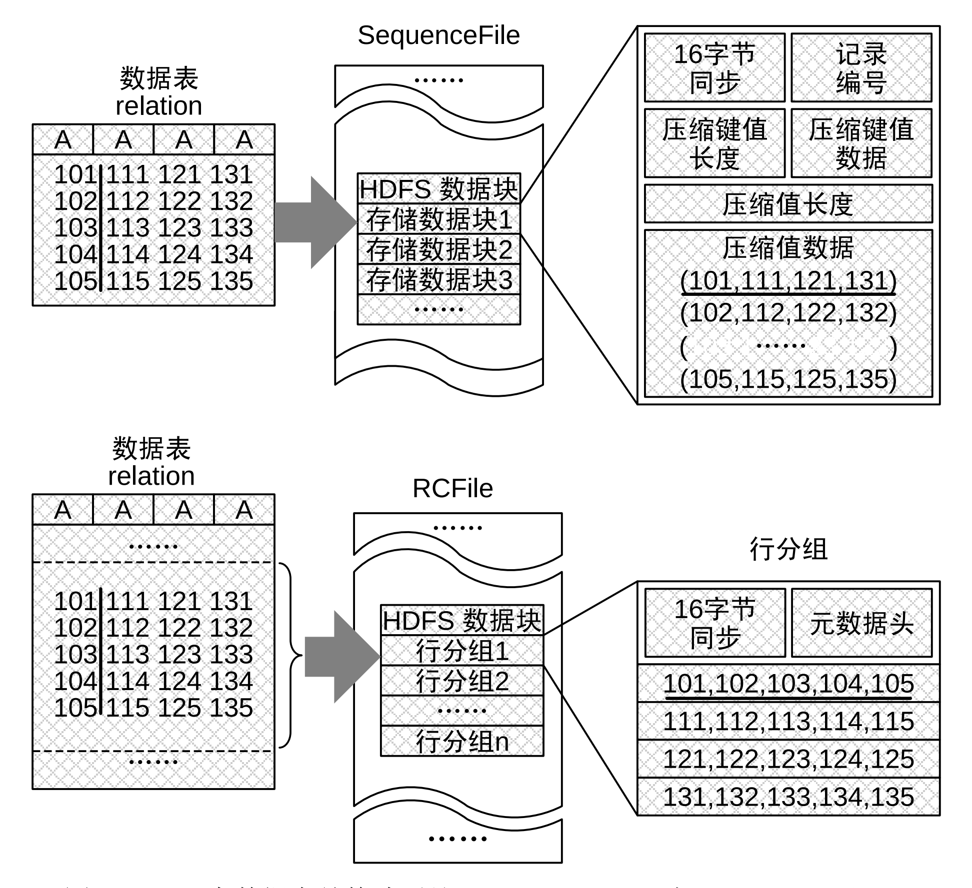


图5. Hive 中数据存储格式对比，SequenceFile（上）RCFile（下）

我们在 Hadoop（版本 0.20.1）和 Hive 配套版本上对 RCFile 进行测试。测试环境：6 节点，每节点配置为：6GB 内存、2 路双核 AMD Opteron(tm) 2000 MHz CPU、Linux 2.6.18-128.el5 x86_64running CentOS release 5、千兆以太网互联。实验结果表明，相对于原来的 SequenceFile 格式，RCFile 可以节省 25%的存储空间，并且性能也提高 30%。

RCFile 的实现代码现已贡献给 Hive 开源项目，并已经应用于 Facebook 公司的 Hadoop 生产系统。经 Facebook 公司实测可节约存储空间 25%。与 Apache Hive 数据仓库系统之前缺省使用的行存储技术 (SequenceFile) 相比，RCFile 在不影响查询性能的前提下节省高达 20% 的磁盘空间，与雅虎公司开发的数据分析系统 (Apache Pig) 中的列组存储技术相比，RCFile 在磁盘利用率相当的情况下可以将数据加载性能提高 23% 左右。自 0.4.0 版开始，RCFile 已经集成到 Apache Hive，用以替换 SequenceFile 成为缺省的二进制数据存储结构。据了解，从 2009 年起，国际上和中国境内使用 Apache Hive 的很多互联网公司逐步转向使用 RCFile 存储数据。RCFile 已经成为诸如 Apache Hive 的分布式离线数据分析系统中数据存储结构的事实标准。

4.2 互补式聚簇索引技术

随着网络应用数据量的不断增大，为了满足高读写性能、低存储开销和高可靠性的要求。谷歌提出了以 BigTable 为代表的列存储结构，其特点是数据按主键顺序存储，同时数据又按主键被分片到大量数据结点处理,从而为各种应用提供海量数据上低响应时间、高吞吐量的数据存取服务。

HBase 是 Hadoop 中 BigTable 的一个开源实现，是一种适用于海量数据（TB 到 PB 级）下单个维度（仅仅是主键）区间查询的数据库系统。HBase 可以按主键迅速定位数据，同时还支持主键上高吞吐量的范围查询。但是实际应用中往往要将数据按多个不同的属性进行排序以支持多个维度的区间查询。目前 HBase 中还没能提供一种查询速度快、存储开销低的索引方法来实现以上功能。

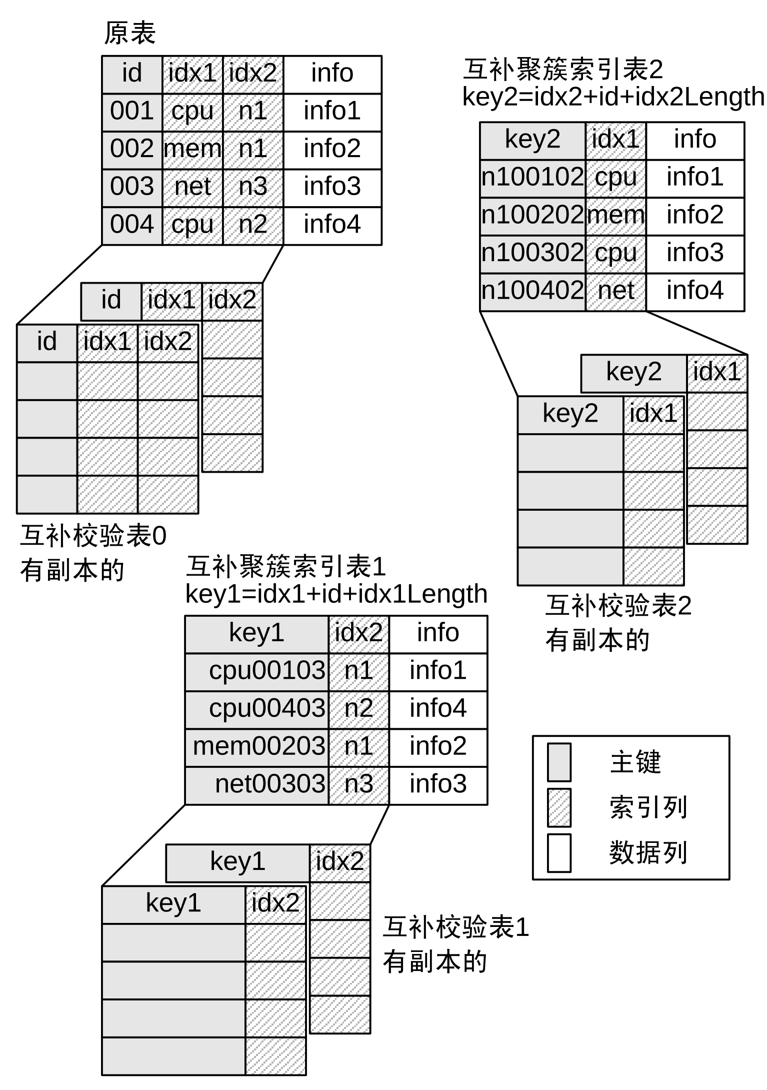


图6. CCIndex 数据表逻辑结构示意图

为了满足海量数据上的多维区间检索的需求，我们基于 HBase 实现了互补聚簇索引方法和系统。提供多个索引表以对需要检索的属性列按序存储（互补），并在索引表中存储所有的记录数据（聚簇）。这样在进行多维区间查询时，就可以直接从索引表中取得数据，从而大大提高查询速度。CCIndex 数据表逻辑结构见图 6。同时 CCIndex 方法屏蔽了 HBase 底层的数据备份，使用索引表中保存的数据做记录级的数据恢复，保证了少量增加存储开销的同时大幅提高查询速度。

CCIndex 方法具体包括以下三方面内容：

– 数据组织：CCIndex 方法把用于备份数据的副本组织成为多份互为补充和校验的互补聚簇索引表，利用索引表上高效的连续扫描代替原表上的随机读取，从而大幅提高多维区间查询性能。

- 查询处理和优化：CCIndex 方法首先将查询串转换为查询计划树，对查询语句进行去重合并等简单优化。然后基于 HBase 的分片信息对子查询的结果集的大小进行估算，最后挑选最小子查询在对应的聚簇索引表上执行查询过程。
- 数据恢复：CCIndex 在互补校验表中保存各索引表数据主键的对应关系，通过互补聚簇索引表和互补校验表进行数据增量恢复，保证了与通过数据副本进行数据恢复时相同的可恢复性同时仅少量增加存储开销。

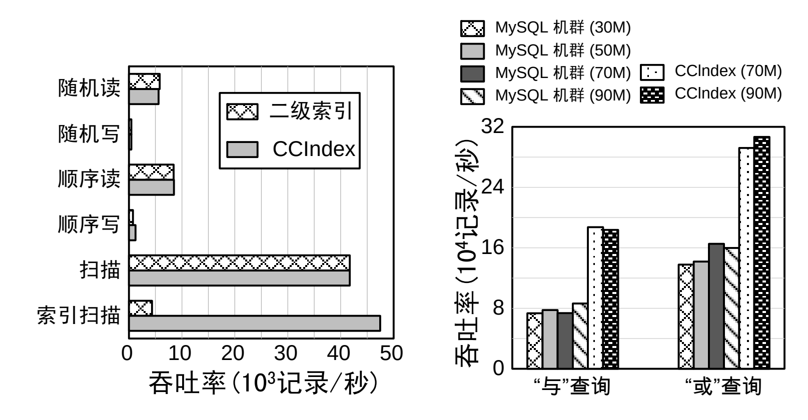


图7. CCIndex 的微基准测试（左）与综合测试（右）结果

在微基准测试中，CCIndex 与 IndexTable（HBase 中二级索引方法的实现）相比，索引查询速度提高 11.4 倍，顺序写和随机写的速度分别提高 1.5 倍和 2.2 倍。微基准测试环境由 3 个节点组成，每节点包含两颗双核 CPU，2000 MHz dual-cores AMD Opteron^{TM} Processor 270，6 GB 内存，321 GB RAID5 SCSI 硬盘，千兆以太网互联，操作系统为 Linux 2.6.18-128.el5 x86_64running CentOS release 5，测试数据集为 100 万条长度为 1KB 的随机数。

在综合测试中，我们将监控应用 Nagios 产生的数据按"时间"、"CPU 负载"、"网络流量"和"集群号"四个属性建立索引。在实际应用中比较 CCIndex 与 MySQL Cluster 区间查询时的吞吐量。综合测试环境由 16 个节点组成，总计 64 核，96GB 内存，其中单个节点配置与微基准测试一致，除了改用 186GB RAID1 SCSI 磁盘。数据集为集群监控系统产生的 1.2 亿条数据，单条记录的平均长度为 118 字节。

实验结果表明执行以 OR（"或"）连缀的区间查询时 CCIndex 的吞吐量为 MySQL Cluster 的 1.9 倍。执行以 AND（"与"）连缀的区间查询时，CCIndex 的吞吐量为 MySQL Cluster 的 2.1 倍。

查询举例：
```
OR: CPU Utilization > 0.8 or CPU Utilization < 0.3 or
    ClusterID > 3。
AND:CPU Utilization >0.3 and CPU Utilization <0.9 and Cluster
    ID > 3。
```

目前我们已经在集群系统监控应用中使用了 CCIndex 技术，使监控系统在处理数据规模和速度两方面得到显著提升。结合淘宝公司"数据魔方"实时数据分析系统的实际需求，作为其全属性实时计算系统的核心，CCIndex 技术经适配和优化后，已集成到生产系统中投入实际运行。CCIndex 增强了全属性实时计算系统的扩展性和性能。目前，系统处理的数据条目超过 100 亿。采用 CCIndex 技术后，在硬件规模保持不变的前提下，系统处理的数据时效范围从原来的 7 天增大到 3 个月，处理容量增大了一个数量级，系统吞吐率增

大了 7 倍，对原来延迟大于 1s 的查询请求响应时间平均降低了 57.4%。

4.3 基于硬件加速的流式透明压缩技术

当前的分布式文件系统一般采用多副本的策略。在大规模机群中，这会带来不可忽略的巨额存储开销。同时，对于分布式文件系统之上的系统或应用，如分布式数据库、分布式数据仓库、MapReduce 框架或其它应用，也有可能会产生冗余数据。这样，会使得数据的膨胀率更高，读写性能瓶颈问题更为凸显，现有的分布式文件系统难以同时满足高性能、高可靠性和低存储开销的需求。

通过对分布式文件系统之上应用的类型进行分析可以知道，这些应用使用或产生的大部分数据是文本信息，特别是离线或在线分析系统中的数据基本都是文本。而文本本身是一种高度可压缩的数据，因此通过引入一种快速的数据压缩方法，可以有效降低数据的存储开销，提高磁盘和网络读写的有效带宽，从而提高应用的吞吐量[9]。

传统的压缩方法，如 GZip，在压缩或解压缩过程中会占用大量的 CPU 资源，使系统的处理能力受到较大的影响。虽然压缩能使系统的存储开销减小，但也有可能会使系统的处理能力下降。随着硬件技术的发展，可以使用硬件设备来压缩数据，达到分流 CPU 负载和提高压缩处理效率的目的。我们的解决方法提供一种分布式文件系统上的基于硬件加速卡的流式透明压缩技术，在占用少量系统资源的情况下，完成对用户透明的压缩和解压缩过程，能够有效降低系统的存储开销和提高系统的处理能力。

Swift 文件系统（SwiftFS）方案首先采用硬件加速卡来对内存缓冲区进行压缩或解压缩。加速卡具有并行处理能力，压缩效率高，压缩或解压缩过程只消耗少量的 CPU 资源。其次，数据的压缩或解压缩对于用户是完全透明的，无论是写入或读取数据，都可以提高磁盘和网络读写的有效带宽。此外，采用分片式压缩格式，将文件分成大小为 64KB 的分片（chunk），能达到较好的压缩效果，而且硬件加速卡只需要很小的缓存。每一个分片，在真实的压缩数据之前是该分片的头部信息，包括：原始数据大小和压缩数据大小。最后，对上层系统提供一个基于硬件加速卡的流式压缩器，用以封装原有的输入流或输出流，创建压缩或解压后的输入流或输出流。如果硬件加速卡出现故障，采用软件压缩/解压缩，形成良好的容错机制。

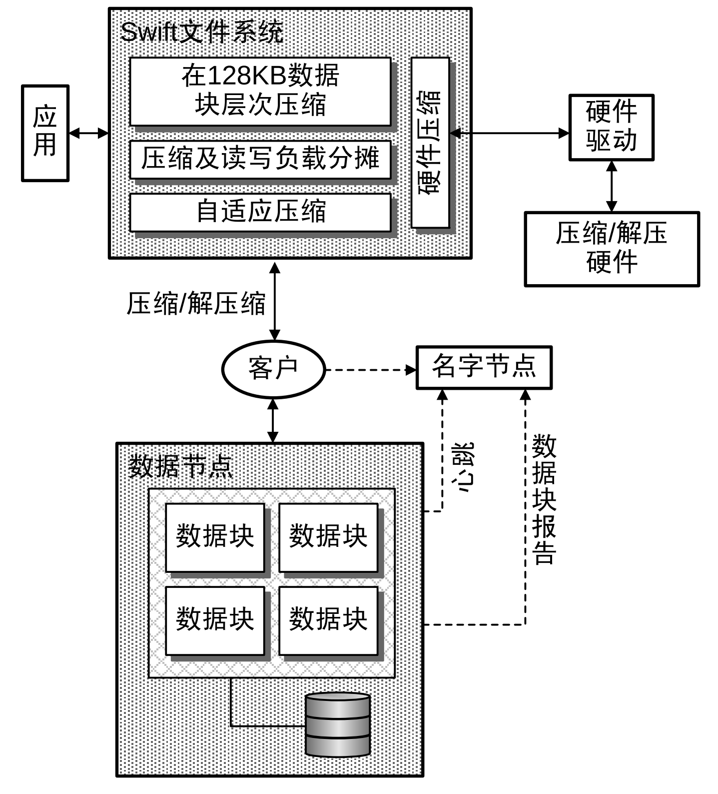


图8. 基于硬件加速卡的流式透明压缩原型的系统结构图

原型的系统结构如图 8 所示。因为 Apache HDFS 是 Google GFS 的开源实现，是 Hadoop 生态圈各组成部分的公共基础，所以我们采用基于 Apache HDFS 实现基于硬件加速卡的流

式透明压缩的方案。HDFS 之上的应用使用 HDFS 客户端来写入或读取文件。默认情况下，每个文件被分成大小为 64MB 的数据块。这些数据块存储在不同的数据节点上，每个数据块有 3 个副本。基于硬件加速卡的流式透明压缩器位于客户端和数据节点之间。客户写入数据时，数据先经过压缩再发送到数据节点；客户读取数据时，先将从数据节点读取的数据进行解压缩后再返回给客户端。HDFS 之上的应用也可以使用独立的基于硬件加速卡的流式透明压缩器对数据进行压缩后再写入到 HDFS 中，通过不同的使用方式来满足不同的需求。

借助硬件加速卡，压缩过程只占用少量 CPU 资源，能够分流达 20%~30%的 CPU 负载，压缩处理吞吐率高于磁盘读写带宽。从目前测试效果看来，对于真实的数据，压缩比大约为25%左右，有效降低了存储开销，将磁盘的有效带宽提高了 4 到 5 倍。压缩过程对上层应用透明，因此基于 HDFS 的在线或离线数据分析系统（HBase、Hive 等）都可以方便地使用。该项技术已经随天玑大数据处理一体机广泛应用到政府、国防、安全和公安等部门，大大提高了海量数据处理的计算效率，同时节省了存储空间。

5 天玑大数据处理引擎典型应用

5.1 读写分离统计分析型应用

结合淘宝的数据魔方在线系统的实际需求，作为数据魔方全属性实时计算系统的核心，天玑大数据处理引擎的重要组件—天玑 Base 已上线投入实际运行，使淘宝网原有的业务逻辑能够直接迁移到经改进的 HBase 上，同时增强了全属性实时计算系统的扩展性和性能。目前，该实时计算系统处理的数据记录超过 108 亿。天玑 Base 的索引及分布式查询技术解决了诸如 HBase 等当前主流的列簇式 NoSQL 数据库系统在多列查询上的功能缺失和性能低下的问题，通过融合各种索引技术及联合优化，可以对 NoSQL 中数据非主键列进行定位和查询，从而弥补了 NoSQL 与传统关系数据库相比查询功能及能力的欠缺。同时利用服务端计算技术，可以对海量数据进行本地化聚合计算而无需进行大量数据拷贝传输。如此经过强化的查询统计能力配合 NoSQL 的高扩展性及大吞吐量的数据处理能力，使众多关系型数据库面对的数据处理瓶颈得以克服。

5.2 低延迟流式处理型应用

对于用户流量来源以及用户点击行为的分析一直以来都是淘宝的"量子统计"提供的服务中最为重要的组成部分。以往采用传统技术只能为用户提供按小时统计的分析数据，即用户可查询店铺内某一天的 24 小时分时段的数据报表。其内容包括各时段用户浏览量、访客数及来源和店内浏览路径。而采用天玑 Base 进行数据流式存储和统计之后，店主可以实时地看到当前正在浏览客户的实时点击行为。新系统实时地收集分析了淘宝全网用户点击日志，统计内容包括淘宝 300 万店铺的实时 UV[9]、PV[10]值，并能绘制出淘宝网日均 1.2 亿用户的实时点击行为图示，最后将这些信息分类推送给相关店主。整个系统的数据处理延时仅为 2 至 3 秒。实际日志处理量为 3 万至 5 万记录每秒，每天 20 亿记录，数据写入操作为 15 至 25 万次每秒，单日原始数据量为 600GB，存储一周用户数据则原始数据量为 4TB 左右。

5.3 大并发访问型应用

腾讯网是目前中国最大的互联网综合服务提供商，也是中国服务用户最多的互联网企业之一。截至 2011 年 9 月 30 日，QQ 即时通信的活跃账户数达到 7.117 亿，最高同时在线帐

[9] unique visitor 访问某个站点或点击某条新闻的来自不同 IP 地址的人数

[10] page view,页面浏览量，或点击量

户数达到 1.454 亿。其数据平台一直致力于发掘用户数据的价值，为用户提供更为精准的个性化服务。广点通即数据平台核心产品之一，旨在根据用户访问数据提高平台广告推送效率。

面向海量用户访问数据的实时存储查询系统是广点通智能推荐系统的基础。全内存分布式的天玑 Base 优化了线上系统查询性能，提高了存储层数据访问效率，大幅减轻集群内部网络压力，提高了广点通整体性能，并成功支持了对存储性能要求更高的复杂用户推荐算法。新系统经过相应优化之后，实测单机查询性能提升 20 倍，占用服务器数量缩减为原系统的 1/5，日均处理日志数量 30 亿记录，处理用户请求数量达 25 亿次。

6 结语

Hadoop 是大数据计算领域的一项具体技术，一套软件系统和工具。因其开源而对推动大数据计算技术发展起到了重要作用。面向不同的应用需求，基于 Hadoop 的数据处理工具也应运而生。天玑大数据处理引擎集成了 Hadoop 生态环境中成熟且社区活跃的组件，如 Hive、HBase 等，并整合了天玑团队的众多研究成果，如 RCFile、CCIndex、SwiftFS 等，可以满足舆情分析、社会计算、商业智能和数据挖掘等大数据处理的实际需求。可以预见，大数据计算的出现将催生更多、更好、更面向大众的新应用，而新应用的出现更能够加快大数据计算技术发展的步伐。

参考文献：

[1] Jeffrey Dean and Sanjay Ghemawat.. MapReduce: Simplified Data Processing on Large Clusters, Communications of the ACM, vol. 51, no. 1 (2008), pp. 107-113.

[2] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google File System, in Proceedings of the 19th ACM Symposium on Operating Systems Principles, ACM, Bolton Landing, NY (2003), pp. 20-43.

[3] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A Distributed Storage System for Structured Data, In Proceedings of OSDI 2006, pp. 205-218, Seattle, WA, 2006.

[4] Apache Hadoop. http://hadoop.apache.org/.

[5] Apache Hive. http://hive.apache.org/.

[6] Apache HBase. http://hbase.apache.org/.

[7] Yongqiang He, Rubao Lee, Yin Huai, Zheng Shao, Namit Jain, Xiaodong Zhang, and Zhiwei Xu. “RCFile: a fast and space-efficient data placement structure in MapReducebased warehouse systems”, in Proceedings of International Conference on Data Engineering (ICDE 2011), pp. 1199-1208, Hannover, Germany, 2011.

[8] Yongqiang Zou, Jia Liu, Shicai Wang, Li Zha, Zhiwei Xu. CCIndex a complemental clustering index on distributed ordered tables for multi-dimensional range queries, in Proceedings of the 2010 IFIP international conference on Network and parallel computing, pp. 247-261, Zhengzhou, China, 2010.

[9] Nicolae B, Moise D, Antoniu G, Bouge L, Dorier M. BlobSeer: Bringing high throughput under heavy concurrency to Hadoop Map-Reduce applications. Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on. 2010: 1~11.

作者简介：

查 礼： 中国科学院计算技术研究所，北京 100190 副研究员，char@ict.ac.cn
程学旗： 中国科学院计算技术研究所，北京 100190 主任、研究员，cxq@ict.ac.cn